Universidad Autónoma de Madrid

Escuela Politécnica Superior

Trabajo de Fin de Grado

DESARROLLO DE PLATAFORMA DE APLICACIONES DE AYUDA PARA NIÑOS CON NECESIDADES ESPECIALES SOBRE RASPBERRY PI

Grado **de Ingeniería Informática**

Guillermo Climent González

DESARROLLO DE PLATAFORMA DE APLICACIONES DE AYUDA PARA NIÑOS CON NECESIDADES ESPECIALES SOBRE RASPBERRY PI

AUTOR: Guillermo Climent González
TUTOR: David Camacho Fernando

Resumen

La tecnología está muy presente en el día a día de las personas: smartphones, redes sociales, videojuegos etc. La educación no es una excepción, cada vez se observan más ordenadores y pizarras digitales en las aulas de nuestro país. Se ha pretendido crear un prototipo de plataforma educativa orientada a la enseñanza, especialmente diseñada para la educación especial, aunque extensible a otros ámbitos. Consta de un sistema hardware basado en Raspberry Pi, junto con una interfaz de juego con matrices LED personalizables diseñada para llamar la atención de este tipo concreto de alumnos y mejorar la calidad de la enseñanza. Los juegos ejecutables, pueden ser creados por los padres y los profesores a través de una interfaz WIFI y cualquier dispositivo con un navegador, y no hace falta ningún conocimiento técnico. Con el diseño de la plataforma, se pretende proveer a los educadores de una plataforma simple, barata y mantenible para mejorar la calidad de la enseñanza y hacerla más personalizable, y que de esta forma, los padres sean también protagonistas de la educación de sus hijos.

Palabras clave

Educación especial, enseñanza, plataforma hardware, Raspberry Pi, python, PHP, Web

Abstract

Nowadays, technology is notably present in people's life: smartphones, social networks, videogames, etc. Education is not an exception, as technology evolves, we can see even more computers and interactive whiteboards in the classrooms all over our country. The goal of this proyect, is creating a prototype of an educational platform, designed for special education and even applicable to other ranges. It is built over a hardware system based on Raspberry Pi, completed with a game interface made of several personalizable LED matrices, specially designed for calling student's attention. These games, can be made by parents and teachers, using a WIFI interface accesible from any device equipped with a web browser, and it is not necessary to have any tecnical formation to create them. The point of designing this system, is provide the teachers with a simple, cheap and mantenible platform to produce a better teaching experience, and enabling the parents to be the main actors in their child's education.

Keywords

Special education, teaching, hardware platform, Raspberry Pi, python, PHP, Web

Agradecimientos

Este proyecto está dedicado a mi familia, en concreto a mis padres y a Noemi por aguantar mi mal humor en momentos difíciles y especialmente a Manuel, quien ha servido de inspiración y de fuente de sabiduría para el desarrollo de esta maravillosa idea. Por último animar a todo aquel que lea este documento a disfrutarlo y agradecer su interés por él.

Índice general

Índice de figuras

1. Introducción

Actualmente, la informática está muy presente en la vida diaria de las personas, especialmente en los últimos 10 años, con la llegada de tecnologías como los smartphones, televisiones inteligentes, consolas de videojuegos de última generación o las redes sociales. Con estos avances, el día a día es más fácil y satisfactorio, tanto en el ocio (organización temporal, mensajería instantánea, videojuegos etc.) como en la vida profesional (organización y compartición de documentos, trabajo colaborativo, gestión de gran cantidad de datos etc.)

Estas tecnologías están presentes también en los centros de educación, donde se usan para proveer a los alumnos de una enseñanza más rica y eficaz. Ya es común encontrar proyectores y pizarras inteligentes en muchas de las aulas de nuestro país, pero existe un área donde las herramientas y técnicas utilizadas para la educación, son más concretas y requieren de más cuidado y dedicación: la educación especial.

Siempre ha sido una gran preocupación para los educadores dotarse de nuevas herramientas para desempeñar sus funciones, un caso especial son los educadores de niños con dificultades especiales, quienes, por la situación de sus alumnos, requieren cada día más recursos para estimular la imaginación de los niños y jóvenes, y empujarles a aprender para mejorar su calidad de vida y la de sus familias

La tecnología ha jugado una importante baza en estos nuevos recursos, ya que elementos como por ejemplo tablets, móviles inteligentes o pantallas de ordenador son artículos que llaman mucho la atención a este tipo de alumnos.

Sin embargo, para los educadores, este tipo de recursos son limitados por varias razones:

- En muchos de los casos, dicho equipamiento es caro y delicado. No muchas familias pueden permitirse comprarlo y especialmente con los alumnos de poca edad, un descuido puede producir que el equipamiento se dañe.

- Las aplicaciones disponibles para estas plataformas son limitadas. No hay muchas empresas o personas que inviertan en software dirigido a este tipo de público y la gran mayoría de los educadores no tienen conocimientos de programación o desarrollo suficientes para construir sus propias aplicaciones de acuerdo a las necesidades de cada alumno.

- Muchos de los recursos educativos convencionales de los que disponen los educadores como por ejemplo murales, tarjetas con ilustraciones o juegos de mesa, solo están disponibles en los centros educativos, y si los padres o tutores de los alumnos quieren trabajar en casa, en muchas ocasiones no es posible.

Por todo ello, la aplicación de las nuevas tecnologías en generar soluciones específicas para este colectivo, podría mejorar notablemente la calidad de vida de estos alumnos y sus familias, a la vez que se facilitaría el trabajo de los educadores y se conseguiría una educación más enriquecedora y si los avances marcan la diferencia, podrían extenderse a otros ámbitos de la educación, como la Educación Infantil y Primaria.

1.1 Motivación del proyecto

La motivación principal del proyecto es facilitar la calidad de vida a unas familias que normalmente tienen que prestar especial atención a la educación de sus hijos, y que muchas veces se vuelve difícil. La informática, permite crear soluciones muy buenas en un tiempo razonable y con poca inversión, que realmente pueden ayudar a estas personas.

Otro concepto que también ha sido motivación de este trabajo, es que normalmente los avances en tecnología se centran en una visión de negocio, y en un colectivo tan pequeño como la educación especial no lo hay, por tanto muchas veces carecen de nuevas herramientas comerciales, y este proyecto es una muy buena ocasión para desarrollar algo que de verdad pueda significar una avance.

También la oportunidad de trabajar con un colectivo tan especializado, aporta una buena visión de cómo es desarrollar un proyecto con ayuda de expertos, especialmente con un colectivo de educadores tan comunicativo y abierto a nuevas tecnologías.

1.2 Objetivos del proyecto

El objetivo principal de este trabajo es desarrollar una plataforma hardware acompañada de un software, que facilite a los educadores su trabajo así como la calidad de vida de los alumnos y sus familias. Dicha plataforma debe cubrir unos requisitos:

- Debe ser una plataforma barata, ya que actualmente, dada la situación económica, los centros educativos y las familias no pueden invertir una gran cantidad de dinero en estas tecnologías.

- Debe ser una plataforma personalizable, se busca la enseñanza individualizada, que el educador pueda diseñar un plan de trabajo específico para cada alumno.

- Debe ser autosupervisada: el hecho de poder trabajar con varios alumnos a la vez sin perder la personalización de la enseñanza, es un concepto que podría ser muy beneficioso para los educadores.

- Debe ser extensible, ya que el colectivo de educadores está siempre dispuesto a colaborar, compartir y crear nuevas técnicas y métodos, por tanto, que el dispositivo pueda adaptarse a las circunstancias, es algo indispensable. Incluso que el sistema sea extensible a la Educación Infantil o a Preescolar también es un punto a tener en cuenta sobre todo al pensar en la amortización del material por parte de los centros educativos.

- Debe ser una plataforma sólida, fiable y fácilmente mantenible. Se buscará que el diseño sea resistente a golpes y que las piezas sean fácilmente reemplazables, para que el coste de mantenimiento sea lo más bajo posible.

1.3 Solución propuesta

Una vez identificados los problemas a solucionar y los requisitos que cumplir, se concluye que la mejor forma de formalizar los objetivos y la motivación del proyecto, es el desarrollo de una consola de videojuegos basados en *templates* o modelos de aplicación.

La plataforma, se basa en Raspberry Pi, un miniordenador programable capaz de ejecutar Linux y ser conectado mediante HDMI o video compuesto/euroconector a cualquier televisor moderno o antiguo, de manera que pueda ser usado como cualquier consola comercial. Para que el sistema sea vistoso, se completa con una interfaz de botones, formados por una matriz de 8x8 LEDs que pueden encenderse de color verde, amarillo o rojo y mostrando dibujos de manera dinámica.

Las aplicaciones constan de pantallas de distintos tipos, como una pregunta, o una relación de conceptos, que van avanzado según se completan. Se provee de un API que permite programar nuevos módulos de pantallas que pueden ser importadas al sistema de manera muy sencilla.

Las aplicaciones se crean a través de un editor web residente en la Raspberry, que es accesible a través de un punto de acceso WIFI generado por la consola, a la que se puede conectar cualquier dispositivo como un smartphone, una tablet o un ordenador. El editor es sencillo y usable, permite importar y exportar aplicaciones y dispone de un banco de recursos (imágenes, sonidos, fuentes etc). También se provee de un API para la programación del editor para nuevas pantallas.

1.4 Público del proyecto

El proyecto presenta una solución amplia para un público diverso:

- **Padres**: El proyecto busca que los padres puedan participar de una forma más activa en la educación de sus hijos, ya que pueden aportar mucho debido a que son los mejores conocedores de los gustos y problemas de sus hijos. También se busca que al poder trabajar los niños en casa de forma más autónoma, los padres tengan algo más de tiempo libre y menos estrés.

- **Educadores**: El proyecto pretende dar a los educadores una herramienta colaborativa, con la que puedan hacer aplicaciones personalizadas para sus alumnos, pero con recursos que puedan ser compartidos y mejorados por todos, de una manera rápida y sencilla.

- **Usuarios avanzados y la comunidad**: Con la estructura de proyecto que se presenta, a través de una API, todo aquel que quiera colaborar y añadir funcionalidades al sistema podrá hacerlo.

2. Estado del arte

En este apartado, se realiza un estudio de la situación actual de las tecnologías relacionadas con el proyecto, así como la explicación de porqué se optó por dichas tecnologías concretas, para el desarrollo de la plataforma, y de cómo estas son capaces de solucionar en gran medida los problemas puestos en evidencia en la motivación del proyecto.

2.1 Funcionalidad: La educación especial

Actualmente, los educadores utilizan multitud de técnicas en su trabajo, y el ámbito de la educación especial es sin duda un área de estudio. Muchas de las técnicas utilizadas, derivan de un programa de investigación creado en la Universidad de Carolina del Norte en EEUU, denominado Treatment and Education of Autistic and Related Communication Handicapped Children (TEACCH), orientado a los niños con trastornos del entorno del Autismo.

El programa fue desarrollado por Eric Schopler y Robert Reichler entre otros, a partir de la tesis doctoral de Schopler y un estudio piloto. Su filosofía se basa en considerar el autismo como una forma de vida y no como una enfermedad que curar provocada por errores parentales. Busca construir un programa individualizado a partir de los intereses, necesidades y capacidades de cada paciente, haciéndolo lo más autónomo posible y trabajando en colaboración con la enseñanza y las familias [1].

Los investigadores desarrollaron unos métodos de trabajo englobados en la terapia del comportamiento, aunque no actúan sobre él directamente. Basándose en teorías que sugieren que los problemas de estos pacientes vienen altamente determinados por dificultades de percepción y entendimiento subyacentes, buscan mejorar la base, como por ejemplo, trabajar en que la persona entienda cómo se espera que actúe bajo determinada situación, o enseñarle la forma correcta de expresar sus necesidades para que sean atendidas y mejoren los niveles de socialización [2].

Figura 1: Ejemplo de actividad TEACHH

Figura 2: Ejemplo de actividad TEACHH

Sin embargo, el reto reside en como transmitir la información al alumno. Por ejemplo, las técnicas tradicionales como las explicaciones verbales, no son efectivas para este tipo de personas, y en muchos casos son contraproducentes, ya que pueden no comprender las expresiones del ponente o incluso estar prestando atención a cualquier otra cosa, como el patrón de los labios del maestro, pueden no entender el significado de una sonrisa o un abrazo de felicitación o incluso no saber cuándo algún estímulo o conversación está dirigido a ellos. Por todo eso, se presenta una gran dependencia visual o táctil de la información [3] [4].

Tras estudiar ejemplos de actividades de este tipo [5] se observó que muchas de ellas, tienen una serie de características comunes, la más importante es que estas actividades tienen una estructura física muy trabajada y rígida, ya que al colocación de los elementos, forma parte muchas veces de la propia actividad, ya que la organización espacial es un tema muy trabajado por los educadores. Esta característica, vista desde el punto de vista de la Ingeniería de Software, ofrece la oportunidad de poder diseñar una serie de *templates* o modelos, que mediante la configuración sencilla de los elementos gráficos o sonoros que los componen, puedan generar un gran número de actividades diferentes para trabajar multitud de ámbitos distintos.

2.2 El elemento procesador: Single-board computers

Debido a los requisitos de este proyecto, era necesario encontrar un dispositivo procesador con algunas características concretas. Tendría que tener soporte GPIO para poder controlar los circuitos externos de los botones, así como un soporte multimedia importante, ya que debía poder mostrar imágenes y sonido que es la parte central del proyecto. Por todo ello, se necesitaba un elemento procesador con cierta potencia, pero que también fuera fiable, ya que se quiere diseñar un producto duradero y resistente, que no necesite mantenimiento, puesto que va dirigido a un público no experto y que se ajuste a los requisitos de precio del producto final.

Se analizaron diversas opciones:

- **Microcontroladores:** Los microcontroladores como Arduino u otras placas de desarrollo fueron descartados. Aunque cumplen con creces los requisitos de conectividad, precio y fiablidad, la dificultad de añadirles soporte multimedia hace que sean una solución poco adecuada. También otra razón en contra es que muchos de ellos ejecutan sus propios sistemas operativos, lo que limita mucho los lenguajes y tecnologías que pueden aplicarse. Sin embargo pueden ser de utilidad para controlar de una mejor forma los circuitos externos ayudando a un elemento procesador con un propósito más general.

- **Mini PCs:** En el mercado hay multitud de tipos diferentes de Mini PC, tanto con procesadores Intel Atom como con ARM u otros sistemas. Muchos de ellos cumplen los requisitos multimedia, sin embargo en muchas ocasiones la conectividad a elementos externos se hace a través de un puerto serie u otras interfaces más complejas ya que no suelen venir equipados con un GPIO. También muchos de estos sistemas están diseñados para sistemas empotrados que necesitan mucha más potencia de procesamiento que la necesaria para este proyecto, lo que produce que muchas de las características de estos miniPC no se usen, lo que es importante si se tiene en cuenta que son dispositivos en general caros.

- **Otros dispositivos:** Una vez analizadas las alternativas más frecuentes, se presenta la idea de utilizar otros dispositivos presentes en el mercado que son de reciente aparición. Debido al auge de las tecnologías ARM y Android, se han desarrollado otra serie de dispositivos, a medio camino entre los MiniPC y los microcontroladores, los Single Board Computer.

Los Single Board Computer (SBC) son ordenadores montados totalmente (RAM, procesador, I/O etc.) sobre una misma placa de circuito, por lo que su arquitectura es fija y muy poco extensible, esta característica los hace en general baratos, ya que se eliminan conectores, controladores de bus y otros elementos presentes en ordenadores normales. También se aprovecha dicha característica para hacerlos de un tamaño reducido.

Figura 3: SBC KIM-1

Aunque los SBC existen desde 1976 (un buen ejemplo es el KIM-1 donde puede verse claramente al estructura de SBC) el auge de los smartphones en los últimos años ha producido un avance en la tecnología de los procesadores de bajo consumo como los ARM y en la miniaturización del sistema completo, llegando a los System On a Chip (SoCs). Aprovechando esta tecnología se han diseñado y construido placas basadas en ARM que pueden ser apropiadas para el proyecto, por su precio, tamaño y características: Beaglebone, Raspberry Pi y Cubbieboard.

	Raspberry Pi (ver. B)	CubbieBoard (ver. 1)	Beaglebone
SoC	Broadcom BCM2835	Allwinner A1X	AM3358/9
CPU	ARM11 @ 700MHz	Cortex-A8 @ 1 GHz	Cortex-A8 @ 720 MHz
GPU	Broadcom VideoCore IV @ 250 MHz	Mali-400MP @ 500 MHz	PowerVR SGX530 @ 200 MHz
Memoria	512 MB SDRAM	1 GB DDR3	256 DDR2
Video compuesto	Si	No	No
Ethernet	Si	Si	Si
USB	Si	Si	No
Conector SATA	No	Si	No
HDMI	Si	Si	No
GPIO	Si	Si	Si
I2C	Si	Si	Si
Soporta Linux	Si	Si	Si
Tamaño	85.60 mm × 56 mm	100mm × 60mm	86.40 mm × 53.3 mm
Precio	40 $	59 $	89 $

Tabla 1: Tabla comparativa entre SBCs


Figura 4: Raspberry Pi


Figura 5: Cubbieboard


Figura 6: Beaglebone

La placa Beaglebone fue rápidamente descartada, debido a la complejidad de su uso y a que no dispone de la capacidad multimedia que se necesita en este proyecto.

La Raspberry Pi tiene una interfaz GPIO para la comunicación con elementos externos, aunque no tan completa como la de una placa Arduino o Beaglebone pero suficiente para este proyecto. Es un dispositivo fiable, que ejecuta una distribución de Linux sobre un procesador ARM, lo que permite aplicar gran cantidad de tecnologías sin mucho esfuerzo.

También contiene un procesador gráfico con una salida de vídeo compuesto y otra HDMI así como una salida de audio, lo que permite cubrir los requisitos multimedia con creces y todo ello en un espacio pequeño y a un precio más que asequible. Debido a sus características, existe en Internet una gran comunidad de desarrolladores centrados en este dispositivo concreto, por lo que se puede encontrar mucha bibliografía y documentación al respecto.

Aunque la placa Cubbieboard también cumple con muchos de estos requisitos, la falta de comunidad desarrolladora, su precio y tamaño superiores, el hecho de que trae Android por defecto instalado y la ausencia de salida de video compuesto, hacen que la elección de elemento procesador para este proyecto sea la Raspberry Pi.

2.3 La interfaz de usuario

Actualmente, la interfaz de usuario es un tema muy estudiado en la Ingeniería del Software, ya que las aplicaciones están orientadas a un público muy amplio, la mayoría de las veces sin conocimientos técnicos, por tanto debe ser intuitiva y sencilla para que el uso de la aplicación sea satisfactoria. En este apartado, se expone el estado del arte en materia de interfaces de usuario así como su aportación al proyecto.

2.3.1 Interfaz en la consola

Como se ha comentado en apartados anteriores, en el caso de la educación especial, se utilizan mucho las tablets, los juguetes, las tarjetas con dibujos, en definitiva, todo aquello que luzca, suene o sea palpable llama especial atención a esta clase de niños. Por tanto se buscó la manera de adaptar la interfaz con la que los niños juegan para que fuera vistosa y versátil y así proveer a los educadores de ese extra de atención que buscan con los recursos que usan actualmente.

Existen precedentes como por ejemplo en mando de consolas modernas como la Wii de Nintendo, que incorpora sensores de posición añadiendo nuevas posibilidades al control de juegos a través del movimiento del mando, al igual que los acelerómetros de los smartphones son aprovechados por los desarrolladores en los videojuego, o la vibración de los mandos, iniciada por las consolas PlayStation de Sony, que juega un papel importante en el control de los juegos.

Sin embargo, para este colectivo, las fórmulas comerciales actuales no son de mucha ayuda, debido a que en general, los problemas de coordinación de estos niños no les permiten interactuar mediante movimientos de un objeto, o incluso pulsando botones tan pequeños como los de las consolas actuales. Por todo ello supone un gran reto diseñar una interfaz hardware sencilla, barata y vistosa.

Para lograr llamar la atención, la mejor posibilidad es utilizar el sentido de la vista [6] , por ello se decidió utilizar unas matrices de LED de 8x8 píxeles de 4 estados. Estos píxeles pueden estar apagados o encendidos en rojo, verde o amarillo, lo que provee de la posibilidad de ir animando y creando figuras en los botones para llamar la atención al alumno. También permiten apagarlos o poner figuras menos llamativas para alumnos que se vean alterados por esas animaciones.

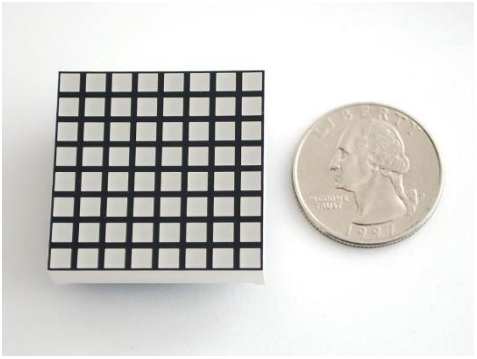


Figura 7: Matriz de LEDs

Figura 8: Marco de botón completo

Las matrices LED irán montadas sobre unos marcos individuales, lo que permite al educador también variar el número de botones para cada alumno de acuerdo a sus capacidades. La forma del marco, permite también tomar los botones con la mano y usarlos como si de pulsadores se tratase, o pegarlos en un mural a mediante velcro o simplemente colocarlos en el suelo o en soportes a ambos lados de la pantalla.

Otra ventaja del diseño de los botones es que su recubrimiento exterior es suave y acolchado haciéndolo resistente a los golpes, construido a partir de gomaeva, una clase de espuma de poliuretano acolchada que puede comprarse en cualquier tienda de manualidades, de forma que pueden construirse en casa, o incluso hacer de la personalización parte del proceso didáctico. Su diseño modular permite también sustituir cualquier componente que se dañe con muy poco esfuerzo y coste.

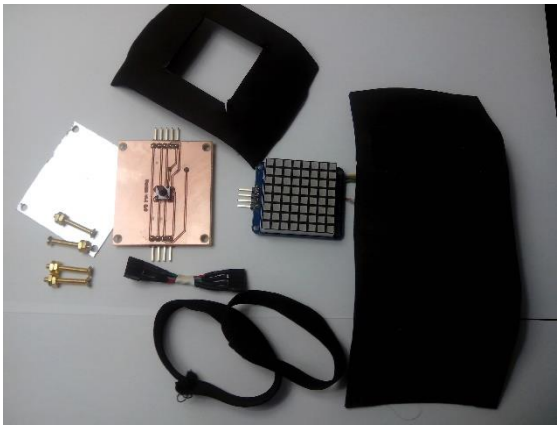


Figura 9: Marco de botón completamente desarmado

Figura 10: Envolturas de botón sin colocar

Aunque los botones individualmente sean vistosos, es necesario desarrollar alguna técnica que permita relacionar qué botón realiza qué acción dentro de la aplicación, y para ello se decidió utilizar una técnica parecida a la que se utiliza en los cajeros automáticos o en muchas máquinas expendedoras de tickets en el transporte público.

Consiste en establecer una relación posicional entre los botones y la pantalla, se colocan los botones en forma de hilera vertical a ambos lados de la pantalla, y en la aplicación, los iconos de acción se colocan de la misma manera, estableciendo así una relación posicional entre un botón y su acción real.

De acuerdo al método TEACCH, los alumnos con necesidades especiales, necesitan tener entornos de trabajo físicamente estructurados [3], y deben aprender a organizar el espacio de arriba abajo y de izquierda a derecha, tal y como normalmente lo organizamos las culturas occidentales, de forma que este esquema de relación posicional es básico para el buen funcionamiento de la plataforma.

Un buen ejemplo de esta colocación, es el modelo que finalmente se decidió construir para probar el proyecto. Consta de una alfombra suave que mantiene la colocación de los botones, y que se coloca frente a la televisión para que el jugador se siente en el suelo frente a ella a jugar.

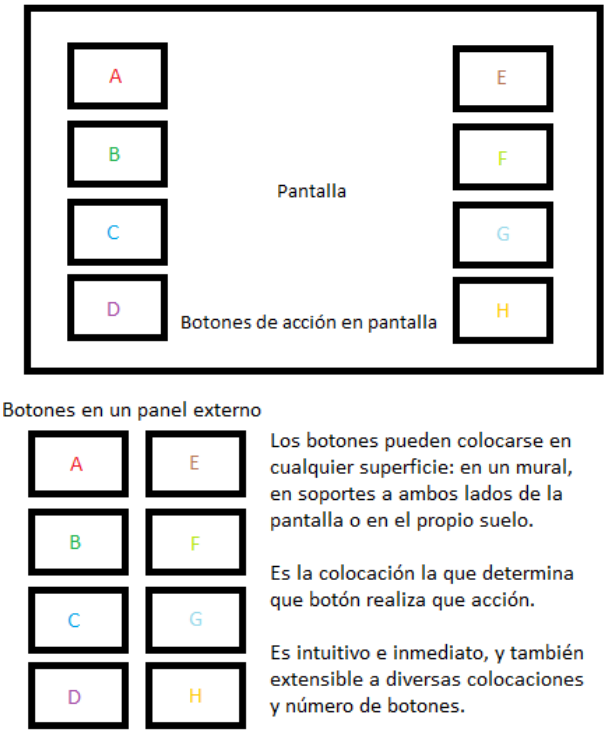
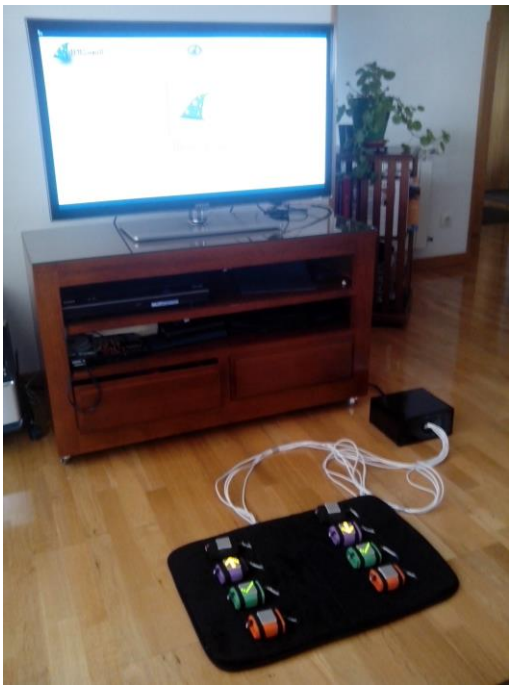


Figura 11: Relación de posición de acciones



2.3.2 Interfaz del editor

Para poder interactuar con la consola y crear las aplicaciones, es necesario implementar algún tipo de editor con el que los educadores puedan crear material. La naturaleza de esta aplicación es crítica, puesto que está dirigida a un público no experto, por tanto debe ser intuitiva y fácil de usar. También se plantea la situación de la plataforma donde esta aplicación debe correr.

Según avanza la tecnología entrando cada vez más en la vida diaria, las interfaces de usuario se hacen más vistosas y minimalistas, y los usuarios más hábiles en su manejo. Un buen ejemplo es el cuidado con el que Apple diseña la interfaz de usuario de sus dispositivos, la importancia de los iconos las tipografías y los colores.

Pero donde realmente ha ocurrido una revolución en el aspecto de la interfaz, es en las páginas web. Impulsado por las redes sociales como Twitter o Facebook, y la aparición del HTML5 los frameworks web hacen cada vez más fácil el trabajo de crear aplicaciones web adecuadas y vistosas. Algunos de estos frameworks son JQuery[1] y Bootstrap[2] (mantenido por Twitter) que junto con PHP y Javascript pueden crear aplicaciones web verdaderamente sorprendentes.

JQuery permite al usuario introducir en la página objetos básicos, como menús seleccionables, pestañas y otros elementos que hacen que la aplicación web sea dinámica y permite al desarrollador mostrar la información de una manera ordenada y sencilla. La fuerte comunidad de desarrolladores de JQuery también provee al desarrollador novato gran cantidad de información así como otros objetos desarrollados por la comunidad, como calendarios, selectores de colores, barras de progreso… que hacen el uso de JQuery muy recomendable si se quiere desarrollar una aplicación web de calidad.

JQuery puede convinarse con Bootstrap, un sistema de estilo mantenido por Twitter, que entre otras cosas predefine estilos CSS para los elementos de JQuery además de añadir nuevos y proveer al desarrollador herramientas para la mejor visualización de su aplicación en dispositivos móviles.

Todo ello produce que hacer del editor de aplicaciones una aplicación web sea una buena idea, si también se combina con un servidor apache en la propia consola y una punto de acceso WIFI que permita a cualquier usuario utilizar cualquier dispositivo con navegador web para conectarse al editor de forma sencilla.

Este paradigma aporta versatilidad y movilidad al proyecto, ya que el editor está integrado en la consola y la interfaz es sencilla y compatible con un gran número de dispositivos.

[1] La documentación de JQuery puede consultarse en http://api.jquery.com/. Para este proyecto se ha utilizado sobre todo su componente JQueryUI para interfaces de usuario.
[2] El API bootstrap puede consultarse en http://getbootstrap.com/

3. La plataforma: Pizard

En este capítulo, se describe el diseño pormenorizado de la plataforma desarrollada en el trabajo. El objetivo es utilizar las tecnologías antes destacadas para desarrollar una plataforma software/hardware que permita a los educadores crear y ejecutar aplicaciones personalizadas orientadas a niños que precisen necesidades de educación especiales, así como el desarrollo de un API que permita extender de una forma sencilla los modelos y la funcionalidad del sistema, con el fin de que la comunidad pueda participar activamente en el desarrollo.

3.1 Arquitectura general de la plataforma

De acuerdo a la naturaleza del sistema, la arquitectura viene determinada en gran manera por la distinción entre la parte hardware y software del proyecto. De esta estructura básica pueden extenderse cada uno de los subsistemas que componen la plataforma.

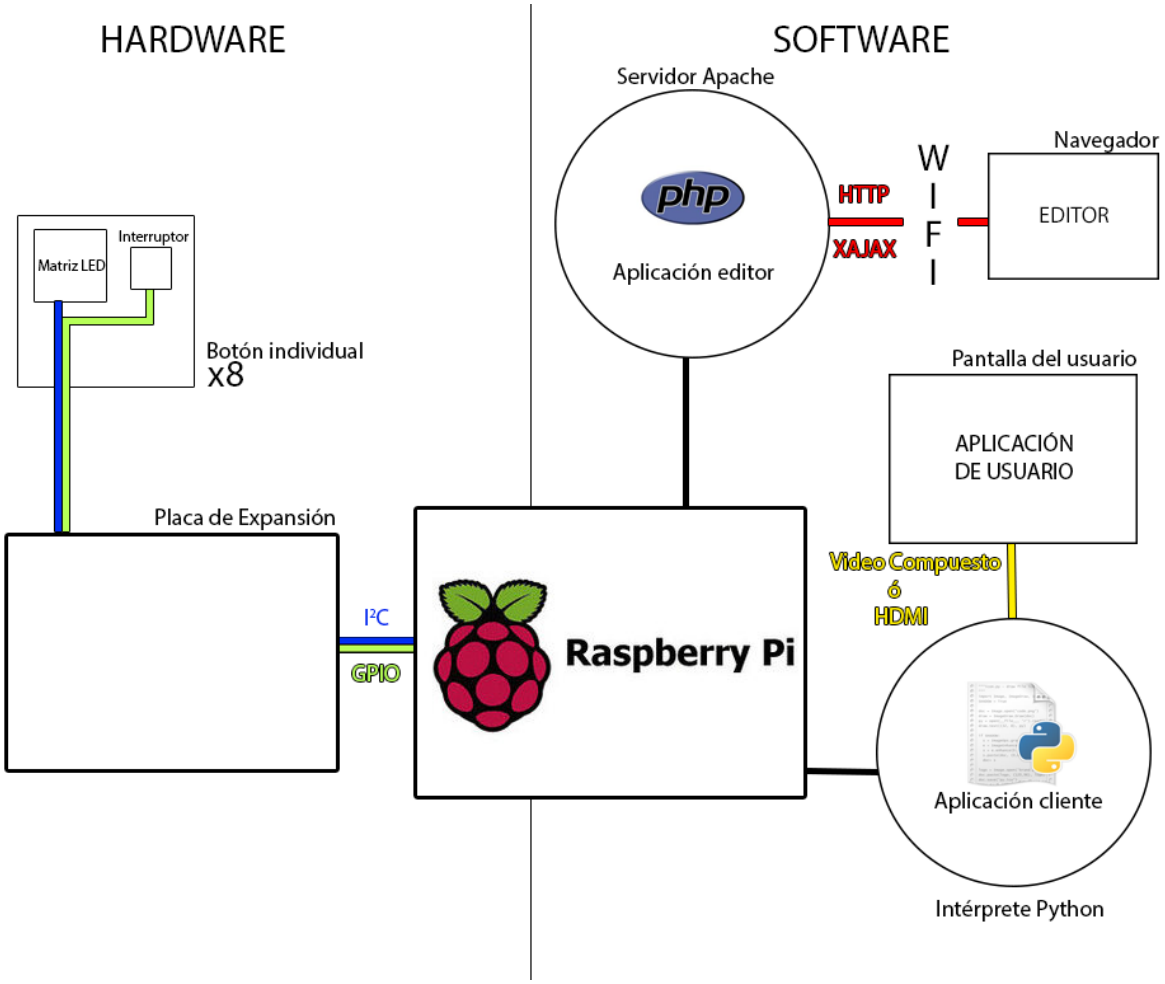


Figura 12: Arquitectura de la plataforma

- **Subsistema Hardware:** Es la parte de la plataforma que controla los 8 botones que componen la interfaz de usuario y los conectan a la Raspberry a través del puerto GPIO y el protocolo I2C para poder ser accedidos mediante el software.

  A su vez se divide en 2 partes diferenciadas:
  - **Los botones individuales:** Tienen como misión mostrar imágenes a través de las matrices LED y recoger las pulsaciones del usuario.
  - **La placa de expansión:** Se encarga de expandir el puerto GPIO e I2C de la Raspberry para generar 8 canales donde conectar los botones. Realiza también funciones de protección de los circuitos de la misma.

- **Subsistema Software I: Aplicación cliente:** Se compone de una aplicación escrita en python que es la encargada de leer las aplicaciones almacenadas y reproducirlas por pantalla. Incluye también un menú de carga de aplicaciones y un API completo para escribir nuevos templates.

- **Subsistema Software II: Aplicación editor:** Consta de una aplicación escrita en PHP que crea una interfaz para la creación de aplicaciones vía navegador. Incluye también un API completo en PHP para la creación de editores para nuevos templates.

- **Subsistema Software III: Sistema Raspbian:** Este sistema consiste en una serie de modificaciones que se deben realizar sobre el sistema Raspbian de la Raspberry para transformarlo en una consola y ejecutar automáticamente todos los procesos necesarios para su funcionamiento, ocultando la interfaz normal de Linux y ahorrando recursos.

3.2 Subsistemas

A continuación se presenta el diseño y el funcionamiento detallado de cada uno de los subsistemas.

3.2.1 Subsistema Hardware

Este subsistema está compuesto por todos los dispositivos físicos necesarios para mostrar las animaciones por las matrices LED de los botones y para recoger las pulsaciones del usuario en las mismas, así como poner a disposición del software una librería con funciones para controlar todos estos aspectos.

3.2.1.1 Herramientas utilizadas

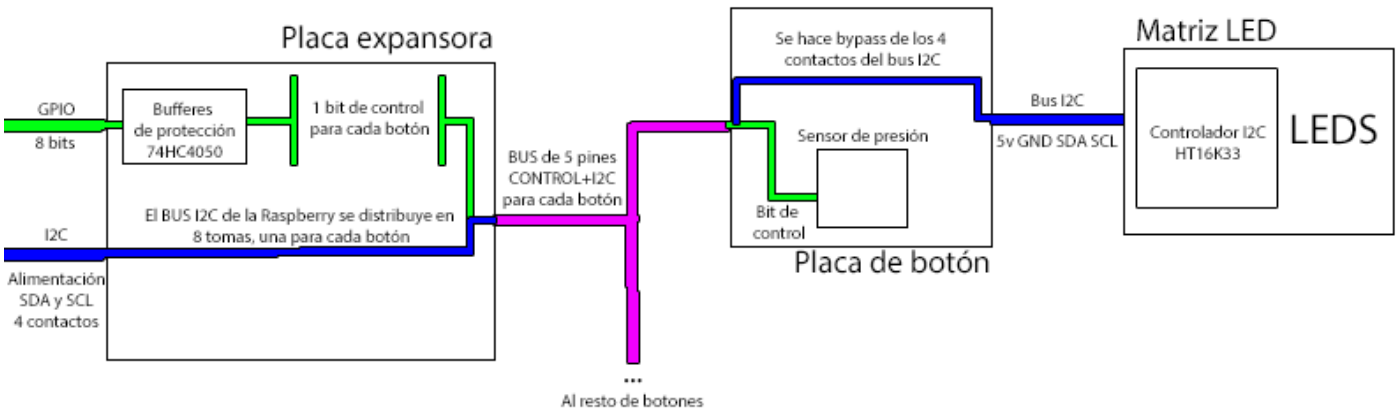
Para realizar todos estos dispositivos, ha sido necesario diseñar y construir placas de circuito impreso (PCB), con el fin de que los circuitos fueran lo más pequeños y fiables posibles. Para ello se ha utilizado la herramienta Altium Designer presente en los ordenadores de los laboratorios de la UAM.

Para el diseño de las PCB, se ha construido una librería, donde se han incluido tanto los componentes para incluir en los esquemáticos, como los footprints del diseño PCB de cada componente incluido en los diseños de las placas. Todos los datos necesarios para la creación de la librería han sido extraídos de las hojas de especificaciones de cada componente.

Una vez realizados los diseños, fueron exportados en formato compatible con la máquina fresadora del taller de circuitos impresos de la EPS y construidos. Todos los esquemáticos de las placas construidas se adjuntan en el anexo de este documento.

3.2.1.2 Funcionamiento lógico

A continuación se presenta un esquemático más detallado del funcionamiento lógico del subsistema:



Las matrices LED[3] están controladas por un chip controlador (HT16K33[4]) con conexión I2C. De forma que para mostrar la imágenes, los chips controladores de todas las matrices de cada botón, deben estar conectadas al mismo bus I2C. Para ello se utiliza la placa expansora como un

[3] Datasheet: http://www.adafruit.com/datasheets/BL-M12A883xx.PDF
[4] Datasheet: http://www.adafruit.com/datasheets/ht16K33v110.pdf

HUB que conecta todos los botones al mismo bus I2C saliente de la Raspberry, mientras que la placa del botón se limita a pasar el bus I2C que le llega hacia el controlador.

Todo el montaje de la matriz LED a partir de la placa del botón, viene suministrada y fabricada por Adafruit Industries[5], también incluye unos pequeños jumpers soldables que permiten la selección de las 8 posibles direcciones I2C para cada botón, por lo que cada módulo de pantalla tendrá soldada una dirección diferente.

Las matrices de LED están alimentadas por la toma de 5 voltios de la Raspberry directamente desde la fuente de alimentación. Las matrices consumen unos 30 mA cada una con la luminosidad al mínimo de acuerdo a la documentación, por lo que se debe usar una fuente de alimentación de al menos 1A para alimentar el sistema, para que haya suficiente para alimentar las 8 matrices, los sistemas de la Raspberry y el WIFI USB. En el caso de disponer de una fuente de mayor potencia, la luminosidad de las matrices es configurable mediante un fichero de configuración. Para ahorrar energía y usar una fuente de menor potencia, puede desconectarse el dispositivo WIFI y acceder al editor mediante el cable Ethernet por red local, o bajar los relojes del sistema (GPU y CPU) desde el fichero de configuración de arranque de la Raspberry, pero el rendimiento del sistema puede verse afectado.

A la placa de expansión también le llegan 8 bits procedentes del puerto GPIO de la Raspberry configurados en modo de entrada. La placa expansora los distribuye a los botones tras pasarlos por unos bufferes (74HC4050[6] conectados al circuito de 3.3 v de la Raspberry) para proteger los circuitos del elemento procesador. Una vez en las placas de los botones, la línea de control de cada botón llega a un interruptor conectado a la toma de 5 voltios y mediante una resistencia a tierra, dando lugar a un circuito del tipo:

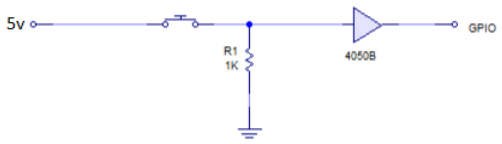


Figura 13: Modelo de circuito de control de presión

Todo ello produce que si hay presión en el botón, lleguen 3.3 voltios al GPIO correspondiente, y 0 voltios si no la hay.

3.2.1.3 Funcionamiento mecánico

El diseño está pensado para ser modular, y poder sustituir cualquier pieza en caso de que se dañe. El botón está formado por el módulo de la matriz LED (controlador I2C y la matriz) suministrada por Adafruit, que se une mediante un cable a la placa central del botón, que está suspendida sobre un rectángulo de aluminio a través de tornillos con muelles, que permiten pulsar el interruptor de membrana que activa el botón apretando la placa PCB contra la de aluminio y que al cesar la presión se vuelva a la posición inicial.

[5] Esquemático del módulo de control suministrado: https://learn.adafruit.com/system/assets/assets/000/008/414/large1024/led_matrix_bicolor.png
[6] Datasheet: http://www.nxp.com/documents/data_sheet/74HC4050_CNV.pdf

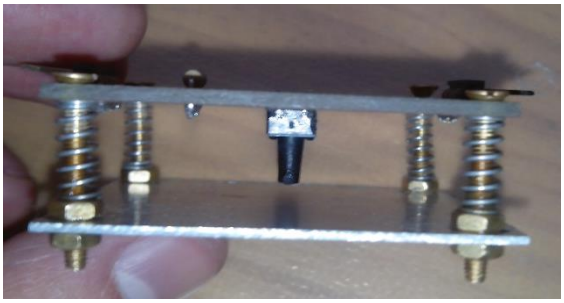


Figura 14: Detalle de la unión entre placas

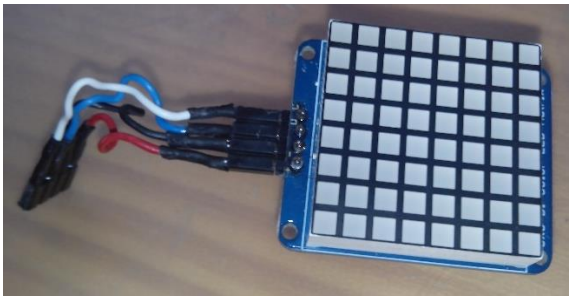


Figura 15: Detalle del módulo de la matriz

La placa distribuidora, está organizada de tal forma que reposa sobre la Raspberry, conectándose directamente a su bus de conexiones por la parte inferior. Por la parte superior se colocan las conexiones para los 8 botones. De esta manera se ahorra espacio y el diseño es más compacto.

Figura 16: Cara superior de la placa distribuidora

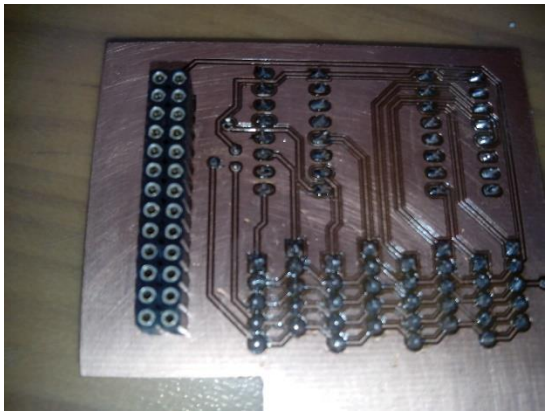


Figura 17: Cara inferior de la placa distribuidora

3.2.1.4 Software de control

Para el control del bus I2C y los botones desde la aplicación cliente se ha creado un módulo *ButtonManager* que ofrece todas las funciones necesarias para controlar los periféricos desde el lenguaje *python*. Está basado en las clases de control de las matrices LED que Adafruit distribuye con sus módulos[7], y para el manejo del I2C, usa la librería *RPi GPIO* de *python*[8].

Incluye clases que representan las animaciones que pueden ser escritas en las matrices, que son leídas desde ficheros JSON con un formato concreto, y son controladas por un hilo encargado de cambiar la figura representada en cada matriz cuando sea necesario. También se incluyen clases que representan el estado de los botones en un tiempo concreto, y permiten conocer cuales estaban pulsados o liberados.

[7] Librerías python para Raspberry Pi: https://github.com/adafruit/Adafruit-Raspberry-Pi-Python-Code
[8] Puede obtenerse en: https://pypi.python.org/pypi/RPi.GPIO

3.2.2 Subsistema software I: Aplicación cliente

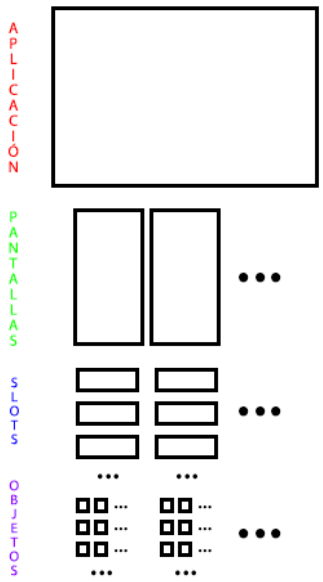
Este subsistema es el que se encarga de leer las aplicaciones del sistema de ficheros, para mostrarlas por pantalla y permitir al jugador interactuar con ellas. Para ello se ha desarrollado un programa escrito en *python* que utiliza la librería gráfica *pygame*[9] para ejecutar las aplicaciones en la Raspberry.

3.2.2.1 Nomenclatura y estructura de las aplicaciones

La aplicación es la unidad funcional del sistema, y se estructura jerárquicamente en diversos elementos. Todos estos módulos se traducen en ficheros JSON en el directorio de la aplicación. La ventaja que suponen los ficheros JSON es que son fácilmente parseables desde prácticamente cualquier idioma de programación, como las aplicaciones deben ser leídas por la aplicación cliente pero también por la aplicación editor (escrita en PHP) un formato común como JSON es fácil de manejar.

- **Aplicación**: Una aplicación se define como un conjunto de pantallas que se suceden una tras otra cuando son resueltas por el usuario.

Figura 18: Jerarquía de objetos de aplicación



- **Pantalla**: La pantalla es la unidad temática de la aplicación, se define como una prueba que el usuario debe resolver para pasar a la siguiente. Cada pantalla está basada en un template o modelo, que define la prueba en sí, y como se supera, por ejemplo, el modelo *Question* consiste en que se le propone una pregunta al usuario con varias respuestas, y no se le permite continuar hasta que haya seleccionado la respuesta correcta. En una misma aplicación, pueden mezclarse pantallas basadas en modelos diferentes.

- **Slots**: Son los componentes de las pantallas, y representan cada uno de los elementos que se muestran en la misma, ya sean texto o imágenes. Su misión es proveer de versatilidad a dichos elementos, estableciendo estados que pueden cambiarse en tiempo de ejecución, para por ejemplo cambiar el gráfico de un elemento cuando es seleccionado. También permite asociar a cada estado con un botón de la interfaz, un sonido o incluso definir una posición diferente con el fin de facilitar el funcionamiento de los modelos.

- **Objetos**: Representan realmente los elementos que aparecen por pantalla, definen sus características gráficas (fondo, borde, tamaño…). Pueden ser de tipo imagen o de tipo texto.

Esta misma jerarquía de objetos, se representa en los directorios de cada aplicación, que se encuentran en la carpeta *content* del proyecto. Se hacen copias locales de todos los elementos

[9] La documentación y la librería pueden descargarse de: http://www.pygame.org

de una aplicación a su directorio, de esta forma importar y exportar aplicaciones puede hacerse de una manera sencilla.

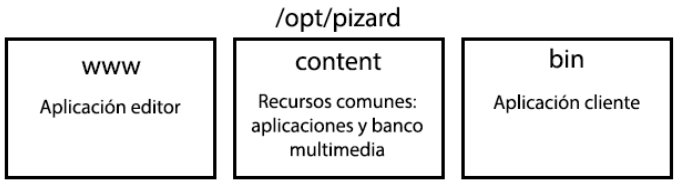


Figura 19: Estructura de ficheros del sistema

De la misma manera, existen módulos *python* con clases que representan también estos elementos, ya que la aplicación cliente tiene que de alguna manera ordenar y manejar estos objetos.

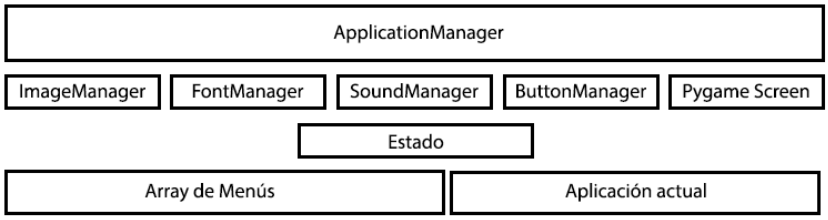


Figura 20: Arquitectura de los manejadores

- **ApplicationManager:** Esta es la clase principal de la aplicación, y la que inicializa y contiene a todos los manejadores y al objeto *screen* de *pygame* que permite controlar la pantalla así como inicializar el log del sistema. Es también la que contiene las clases *menú* que forman la interfaz de usuario de la consola (las pantallas de configuración, selección de aplicación a ejecutar etc.) y el objeto aplicación que está ejecutándose en cada momento. Funciona como una máquina de estados utilizando la estructura de bucle clásica de un videojuego: update-draw-swap screen [7].

  Todas las clases que representen algo que pueda ser pintado por pantalla (aplicaciones, menús, elementos de menú, pantallas, templates, slots y objetos) contienen unos métodos *update* y *draw* que actualizan y pintan los elementos por pantalla respectivamente. Estos métodos en las clases con una jerarquía más alta, llaman a los métodos correspondientes de los elementos que contienen propagando así la orden por todo el sistema.

  Este manejador dependiendo del estado en el que se encuentra, actualiza y pinta un menú o una aplicación concreta según corresponda. También suministra métodos para manejar los estados desde otras clases, para por ejemplo, volver al menú de selección de juego cuando se termina una aplicación.

  Cuando una aplicación se carga en el manejador para ser ejecutada, sobreescribe la anterior y carga sus recursos en los manejadores de recursos que son comunes tanto a los menús como a las aplicaciones. De esta manera solo hay una aplicación completamente cargada en memora a la vez, ya que la Raspberry solo tiene 512 MB de

RAM. Los menús sí que son completamente cargados a memoria, y se mantienen en un array, listos para cambiar de estado en cualquier momento.

- **Manejadores de recursos:** *ImageManager*, *SoundManager* y *FontManager* se encargan de mantener las imágenes sonidos y textos que los objetos necesitan respectivamente. Su funcionamiento es equivalente, y se basa en inicializar los objetos *pygame* necesarios y almacenarlos en un array, devolviendo un identificador y aportando métodos que permitan crearlos, modificarlos y pintarlos en pantalla, ocultando así los objetos *pygame* y los cálculos necesarios a las clases superiores.

De una manera equivalente a como funcionan las aplicaciones, se estructuran también los menús de usuario de la consola, aunque no están compuestos por los mismos objetos que las aplicaciones. Se crearon las clases *ScreenObject* y *ScreenText* que son equivalentes a la clase Object en las aplicaciones, pero no cargan los datos desde JSON sino que se hacen directamente desde el código, obteniendo los recursos también desde un directorio especial y no desde la biblioteca de la aplicación. Estas clases contienen métodos específicos para la creación de botones y menús animados por pantalla.

Debido al gran carácter multimedia y modular del sistema, las aplicaciones están compuestas por muchos ficheros que se organizan jerárquicamente en el disco, para cargarlos es necesario conocer las direcciones de esos ficheros y otras configuraciones, para que fueran accesibles desde cualquier clase en tiempo de ejecución, se produjo la clase *Conf* que parsea un fichero INI con las configuraciones necesarias y lo pone a disposición del programador en forma de instancias mediante un patrón Singleton.

3.2.2.2 Integración de templates

La ventaja de este diseño modular, es que pueden incluirse tantos templates o modelos como se desee, siempre que sigan las reglas de estructura e implementación presentes en la documentación. Los templates son módulos python que son integrados en tiempo de ejecución por la clase *Screen* cuando una pantalla es leída.

Los archivos JSON de una pantalla son diferentes según el tipo al que pertenezcan, pero tienen una serie de campos comunes, como el nombre y el tipo. A partir del tipo de la pantalla, se obtiene el nombre del módulo y de la clase (que deben ser iguales) que determina el funcionamiento de la pantalla (modelo), el objeto *Screen* importa el módulo python determinado por dicho nombre y crea la clase que en él se contenga. De esta manera cuando una orden de actualización o dibujado llega a la pantalla, esta llama a los métodos correspondientes sobre el objeto template creado en la fase de inicialización.

```
# Create the internal template, instantiate the class by name
# All templtes must be in template package and all classes must be named
# after its containing packages
try:
        module = __import__ ("libs.templates."+self._type)
        submodule = getattr(module, "templates")
        classmodule = getattr(submodule, self._type)
        classreference = getattr(classmodule, self._type)
        self._template = classreference (data, app)
except AttributeError as k:
        logger = logging.getLogger('pizard')
        logger.error("Unknown screen type " + self._type)
```

Figura 21: Fragmento de instanciación del template en tiempo de ejecución

Se aprovecha el hecho de que la estructura interna de python son también objetos python que se pueden leer y modificar. Con la función *getattr*[10] y puede obtenerse la clase e instanciarla a partir del módulo importado.

De esta manera, crear un nuevo modelo a efectos de aplicación cliente (luego habrá que crear en el editor las herramientas para poder editar ese tipo de aplicaciones de una manera similar) es simplemente copiar el módulo creado según las especificaciones de la documentación en el directorio de templates. Es posible utilizar todo el API proporcionado por todas las clases manejadoras, además de cualquier módulo o estrategia de python, por lo que el sistema tiene variedad de posibilidades y mucha libertad a la hora de crear modelos.

[10] Puede encontrarse información sobre la estructura interna en tiempo de ejecución de python en: http://es.diveintopython.net/apihelper_getattr.html

3.2.3 Subsistema software II: La aplicación editor

Este subsistema está formado por los elementos que permiten al usuario crear, borrar, importar exportar y editar aplicaciones desde una interfaz web. Este subsistema está escrito en PHP y corre sobre un servidor web Apache instalado en la Raspberry.

3.2.3.1 Tecnologías utilizadas

Durante el diseño de la aplicación editor, se han utilizado diversas tecnologías para producir un código más modular y estructurado:

- Se ha empleado el sistema de templates Smarty[11] para PHP, que permite aplicar el paradigma modelo-vista-controlador, separando la lógica de la aplicación web en los ficheros PHP y la vista (código javascript, CSS y HTML) en ficheros .tpl separados, que se compilan en el servidor antes de que la página sea enviada al cliente.

- Para producir un aspecto visual moderno, se han utilizado las librerías JQuery y Bootstrap, que automatizan mucho el proceso de estilizado de la aplicación, así como algunos objetos desarrollados por la comunidad, como Font-awesome[12] para iconos, *tinycolorpicker*[13] como selector de colores en los formularios y el *File Upload Plugin* de Sebastian Tschan[14] para subir ficheros a través de AJAX.

- Para la creación dinámica de las páginas, se ha utilizado el sistema XAJAX[15].

3.2.3.2 Pantallas del editor

El editor está compuesto por diversas pantallas a través de las cuales el usuario puede interactuar con las aplicaciones del sistema:

- **Pantalla de inicio:** Debido a los recursos de CPU y memoria limitados de la Raspberry, se decidió no ejecutar la aplicación editor y la aplicación cliente a la vez, de forma que se definieron dos estados de uso, el estado de juego donde se ejecuta la aplicación cliente y el estado de edición donde se ejecuta la aplicación editor. El paso de un estado a otro está regulado por la página de inicio del editor, que se encarga de matar o iniciar los procesos pertinentes según corresponda. También informa si los elementos del sistema están funcionando y permite apagar o reiniciar la consola.

[11] Descripción, descarga y documentación en: http://www.smarty.net/
[12] Más información en: http://fortawesome.github.io/Font-Awesome/
[13] Código e información en: http://baijs.com/tinycolorpicker/
[14] Código e información en: http://blueimp.github.io/jQuery-File-Upload/
[15] Documentación del API en: http://www.xajax-project.org/

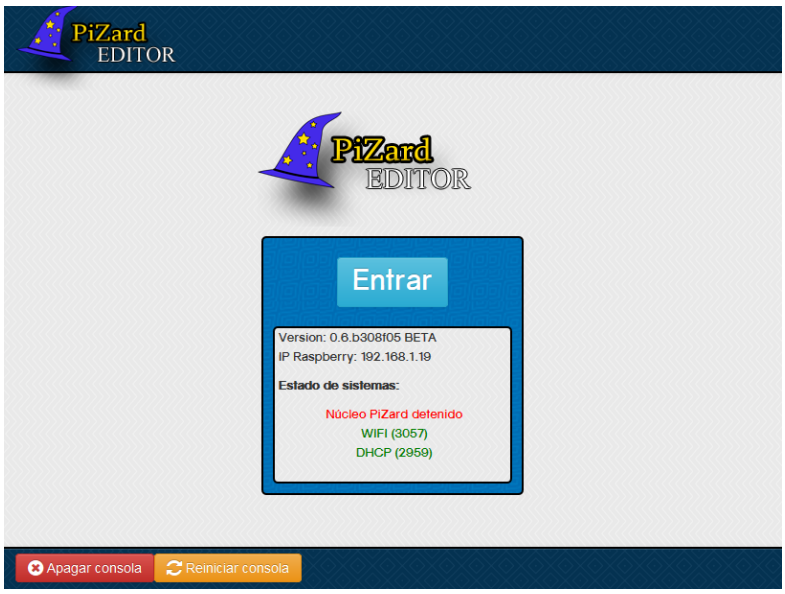


Figura 22: Pantalla de inicio

- **Pantalla de manejo de aplicaciones**: Desde esta pantalla, pueden realizarse todas las acciones sobre las aplicaciones, crearlas, borrarlas, editarlas exportarlas e importarlas:

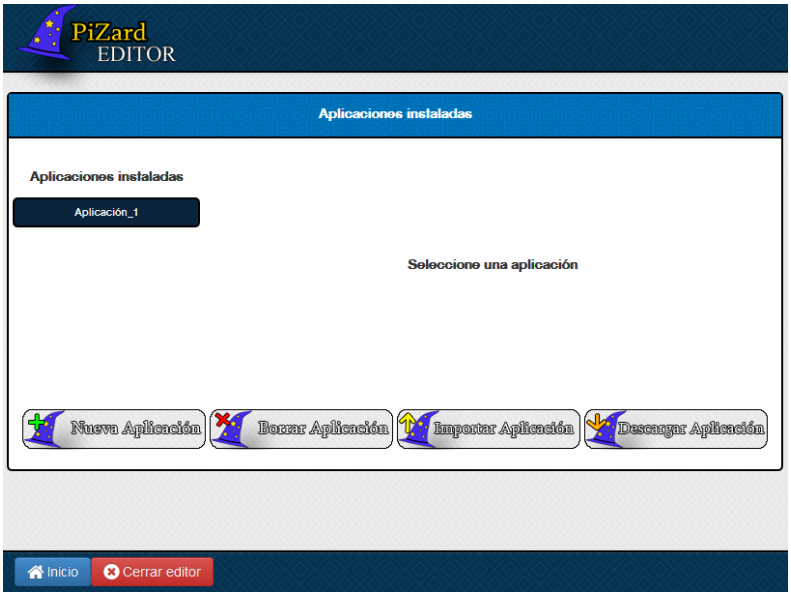


Figura 23: Pantalla de manejo de aplicaciones

- **Pantalla de edición de aplicaciones:** En esta pantalla pueden configurarse las aplicaciones y añadir, borrar y editar las pruebas que las componen.

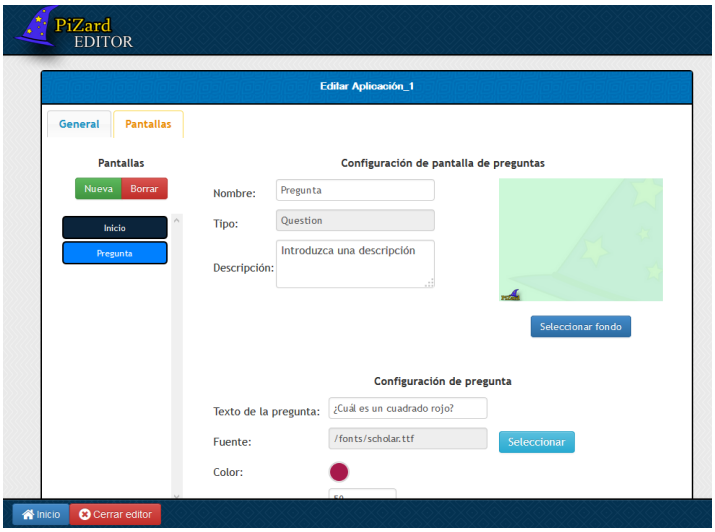


Figura 24: Pantalla de edición de aplicaciones

3.2.3.3 Arquitectura del sistema

Para diseñar la estructura de la aplicación, se ha tenido en cuenta el paradigma modelo-vista-controlador:

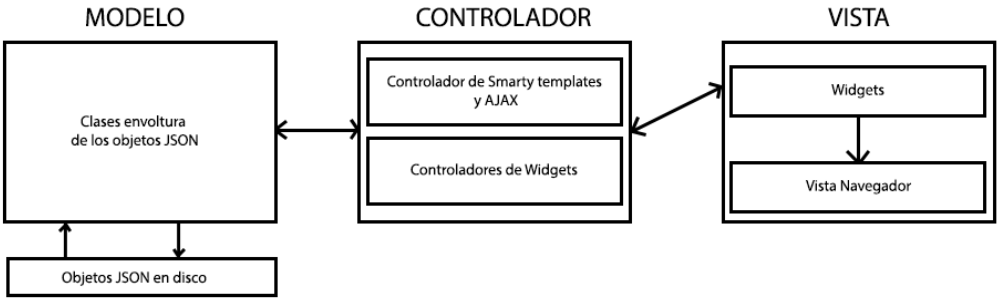


Figura 25: Arquitectura de la aplicación editor

Para facilitar el manejo de los ficheros JSON de las aplicaciones, se han programado clases envoltura que automatizan el proceso de cargar los ficheros JSON, modificarlos y salvar los cambios.

Un tema importante en el diseño, que debe ser muy modular, ya que como en la aplicación cliente, es necesario dar soporte para la creación de nuevos templates, entonces aparece un problema, por ejemplo si se programa un formulario para modificar un objeto, ese fragmento de código debe poder insertarse en cualquier template que necesite modificar alguno de sus objetos.

Para eso, se ha diseñado un sistema de widgets: todos los módulos del sistema son reutilizables y pueden ser insertados en cualquier parte de una página.

Estos widgets están formados por dos piezas, un controlador (un fichero PHP con la lógica de funcionamiento e incursión) y un modelo (un fichero de Smarty templates .tpl). Si se quiere añadir un widget a una página, basta con llamar a un método de incursión desde el fichero PHP de la página e incluir el fichero tpl del widget en el tpl de la página.

Se han desarrollado los siguientes widgets:

- **AppLibrary:** Este widget produce un menú en el que se pueden seleccionar las aplicaciones, ver sus detalles, editarlas borrarlas etc.

Figura 26: Detalle del widget AppLibrary

- **ResLibrary:** Este widget produce un menú desplegable en el que se pueden seleccionar todo tipo de recursos de la biblioteca multimedia del sistema. Es configurable para permitir seleccionar solo cierto tipo de recursos. Devuelve el recurso seleccionado a través de un callback AJAX configurable. Aporta también una vista previa , y permite el borrado así como la importación de nuevos recursos a la biblioteca.

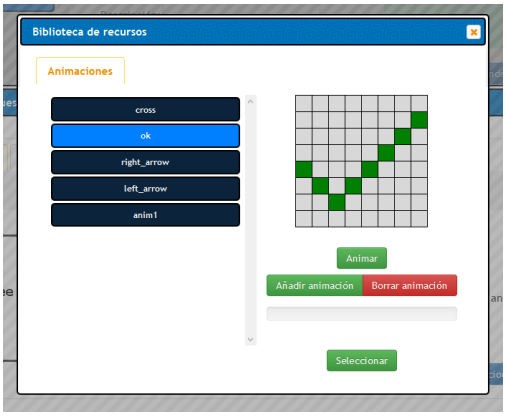


Figura 27: Detalle del widget ResLibrar

- **ObjectEditor:** Con este widget se puede generar un menú desplegable con el que modificar todos los aspectos de un objeto. Maneja la carga desde JSON y su guardado tras realizar cambios. Los campos del objeto están separados en categorías que pueden configurarse y ocultarse para limitar el acceso a los cambios del objeto.

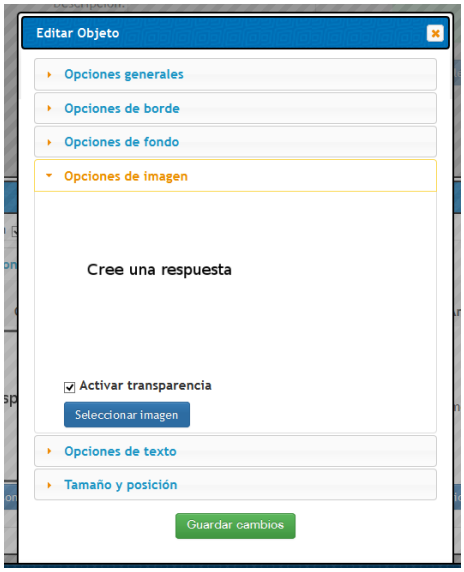


Figura 28: Detalle del widget ObjectEditor

- **EditorWidget: Produce** una ventana donde se permite la edición de una aplicación, desde la creación o borrado de pantallas, hasta el cambio de nombre o imagen.

Figura 29: Fragmento de la vista del widget EditorWidget

Todas las instrucciones concretas de como incluir y utilizar cada uno de estos widgets están en su documentación del API.

3.2.3.4 Integración de templates

De la misma manera que en la aplicación cliente, debe existir alguna forma de permitir a los desarrolladores de templates que han creado un módulo python personalizado, crear un formulario que permita a los usuarios de la aplicación utilizarlo e incluirlo en sus trabajos.

Para ello, se implementa una forma equivalente de inclusión de código a la que se utilizó en la aplicación cliente. La labor del programador consiste en crear el formulario dividido en varios ficheros y colocarlos en el directorio */www/include/templates* para que el sistema los reconozca y los muestre. Los ficheros de los que está compuesto el template son:

- **Un fichero principal** que extiende la clase *Screen* y cuyo trabajo es controlar la lógica de servidor del formulario así como registrar todos los recursos (por ejemplo funciones AJAX) que necesite el formulario, así como interactuar con disco, guardar los cambios o realizar comprobaciones.

- **Un fichero de template HTML** (.tpl) que será incluido en la pestaña de pantallas del editor de aplicaciones cuando se seleccione o cree una pantalla basada en el template.

- **Un fichero con todas las funciones JavaScript** que necesite el formulario. Debe incluir una serie de funciones obligatorias.

- **Una pantalla modelo**, que se copiará al directorio en disco de la aplicación cuando se cree una pantalla nueva. El modelo debe tener todos los parámetros por defecto, tal y como se desee que quede cuando el usuario la cree. Debe ir acompañada de los recursos multimedia que utilice por defecto para que puedan ser copiados a las carpetas locales de la aplicación. Para permitir al programador acceder a la información de la aplicación que contiene a la pantalla, se han establecido unas TAGs que pueden ser incluidas en el HTML del formulario y serán sustituidas por la información apropiada.

Los programadores tienen a su disposición todo el API del sistema, así como los widgets antes explicados y cualquier módulo PHP que precisen.

La información técnica detallada de la creación de templates, se encuentra en la documentación del API que se adjunta con el proyecto.

3.2.4 Subsistema Software III: Sistema Raspbian

3.2.4.1 Modificaciones visuales

Debido a que la consola está dirigida a un público no experto, era esencial construir una capa sobre el Linux que la Raspberry ejecuta para hacer más fácil el manejo del sistema.

Para reducir la carga computacional sobre la CPU de la Raspberry mientras se utiliza la consola se ha modificado el sistema LXDE de escritorio para que no lanzase los módulos visuales, cambiando el fichero */etc/xdg/lxsession/LXDE/autostart* y sustituyendo el lanzador del escritorio gráfico por un lanzador de la aplicación cliente de pizard, de manera que tras cargarse el sistema, se lanza automáticamente la aplicación cliente.

También es algo incómodo que en la fase de carga del sistema tras el encendido, aparezca el log del sistema por pantalla mostrando la inicialización de los diferentes módulos. Para solucionarlo, se utilizó el programa *fbi* (muestra imágenes por pantalla) y se insertó un servicio en */etc/init.d* Para que este servicio se ejecute el primero, se aprovecha el hecho de que los servicios se ejecutan por orden alfabético en la inicialización. De esta forma una imagen permanece en pantalla ocultando el log del sistema hasta que arranca LXDE.

3.2.4.2 Servicios instalados

El hotspot WIFI que crea la consola, es un punto importante, puesto que es la forma más directa de entrar en el editor y poder crear aplicaciones. Para poder establecerlo, se han instalado dos servicios nuevos:

- **Hostapd**: Es el encargado de gestionar el punto de acceso inalámbrico. Gestiona las conexiones y las características de la conexión, como el ESSID, el canal de difusión, las características de encriptado etc.

- **Udhcpd**: Es un servidor DHCP de bajo consumo que se encarga de asignar direcciones IP a los dispositivos conectados al punto de acceso inalámbrico.

Como es lógico, también ha sido necesario instalar y configurar un servidor apache con soporte PHP, para poder ejecutar la aplicación editor.

4. Templates

Para el método TEACCH, la colocación de los elementos es muy importante, y los diferentes tipos de actividades comparten en general una estructura espacial determinada, como se ha explicado en apartados anteriores, por tanto se pueden definir unos pocos modelos fijos que pueden servir para realizar gran número de aplicaciones.

En todos los templates hay que tener en cuenta una serie de requisitos que todas las actividades deben cumplir[16]:

- El espacio de trabajo, debe estructurarse de manera rígida y de izquierda a derecha y de arriba abajo.

- Debe implementarse el aprendizaje sin error. En caso que el alumno falle al realizar una actividad , no se debe continuar hasta que no supere la prueba.

- Debe poder ofrecerse un refuerzo positivo en caso de que el alumno resuelva una prueba con éxito, como por ejemplo un sonido conocido o un pictograma, ya que para esta clase de niños, ver una imagen familiar cuando resuelven un problema correctamente es algo muy motivador.

4.1 Template implementado: Template pregunta

Este template ha sido el implementado en este trabajo, y en él se presenta al alumno una pregunta, acompañada o no de una imagen y una serie de hasta 4 respuestas sobre las que escoger la correcta.

El template está estructurado espacialmente de acuerdo a las directrices del sistema TEACCH, los elementos como la pregunta y su imagen, están a la izquierda de la pantalla, ya que son el inicio de la cadena de pensamiento, que llevará a unos productos, las respuestas, que se encuentran a la derecha, de esta forma el alumno se encontrará cómodo con una ordenación del espacio conocida.

Cada una de las respuestas, puede también desactivarse, ya que para algunos alumnos, cuatro respuestas puede ser demasiado. También es posible configurar una imagen y un sonido de felicitación como refuerzo positivo, que aparecen cuando el alumno supera la.

Con este template, se busca trabajar elecciones, basadas en la comprensión lectora o en la asociación de conceptos a través de la imagen de la pregunta. Algunos ejemplos de actividades concretas estudiadas [5] que pueden realizarse son:

- Asociación de la imagen central con su palabra u otra imagen relacionada o con alguna cualidad en común como color, forma, o clase del objeto. Trabajando así las características de los objetos.

[16] Puede encontrarse una buena guía resumida en español de las características de las actividades para este tipo de niños basada en bibliografía especializada en:
http://www.adaptacionescurriculares.com/Autismo%2012%20metodoTEACCH.pdf

- Muestra de alguna acción, proceso o serie y preguntar al alumno cual es la consecuencia o cuál es el objeto que viene a continuación del elemento mostrado de la serie.

- Puede ofrecerse una imagen modelo y preguntar al alumno cuál de las respuestas es idéntica a la primera para, de esta manera, trabajar la atención.

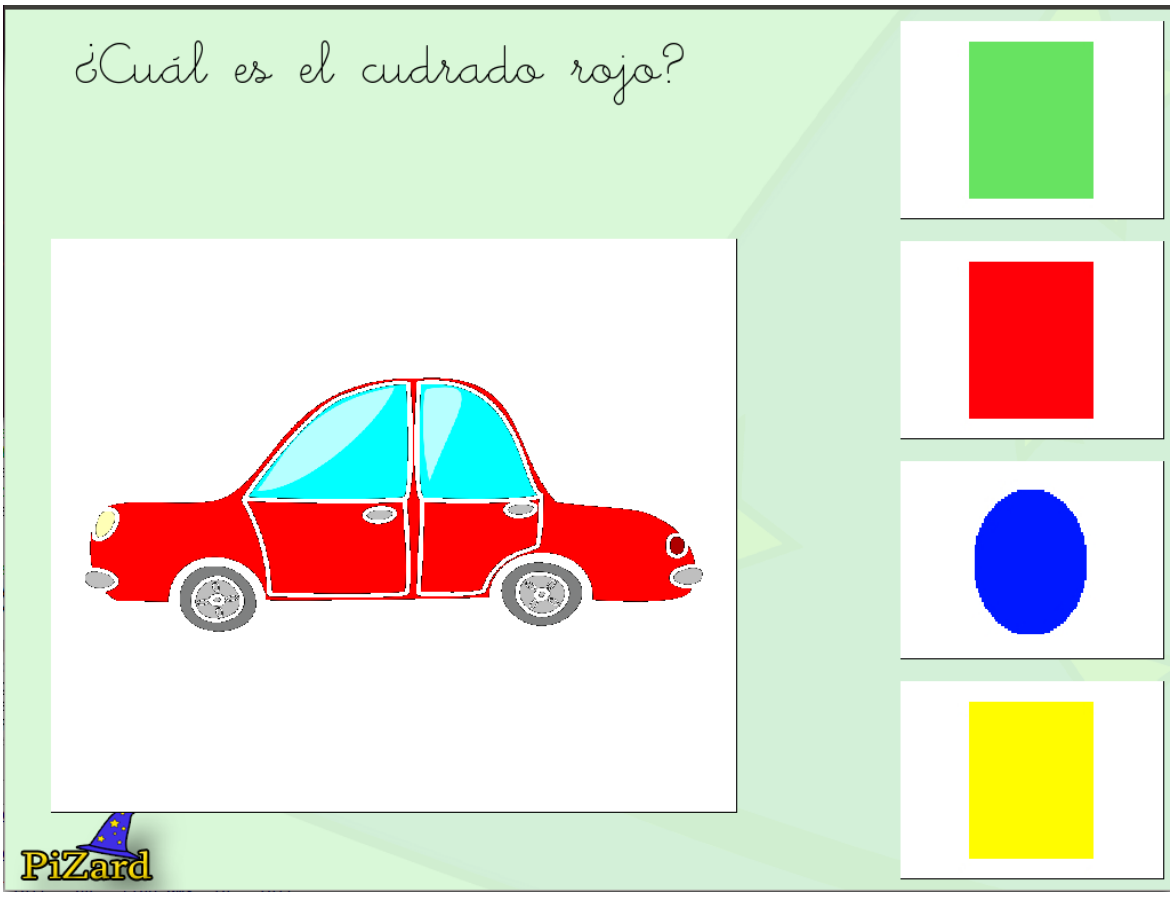


Figura 30: Imagen del template de pregunta

4.2 Template diseñado: Template de asociación o relación

Tras el estudio de las actividades que los maestros realizan con los alumnos de educación especial, se concluyó que otra manera muy efectiva de transmitir información a los alumnos a parte de preguntas simples, es mediante relación de conceptos.

De la misma forma que en el template anterior, se organizan los grupos de conceptos a ambos lados de la pantalla. Cuando el jugador seleccione uno de los conceptos de la izquierda o derecha y a continuación otro del lado opuesto, una línea aparecerá uniéndolos en el caso de que sean correctos. En caso contrario el sistema debe dejar al alumno escoger de nuevo.

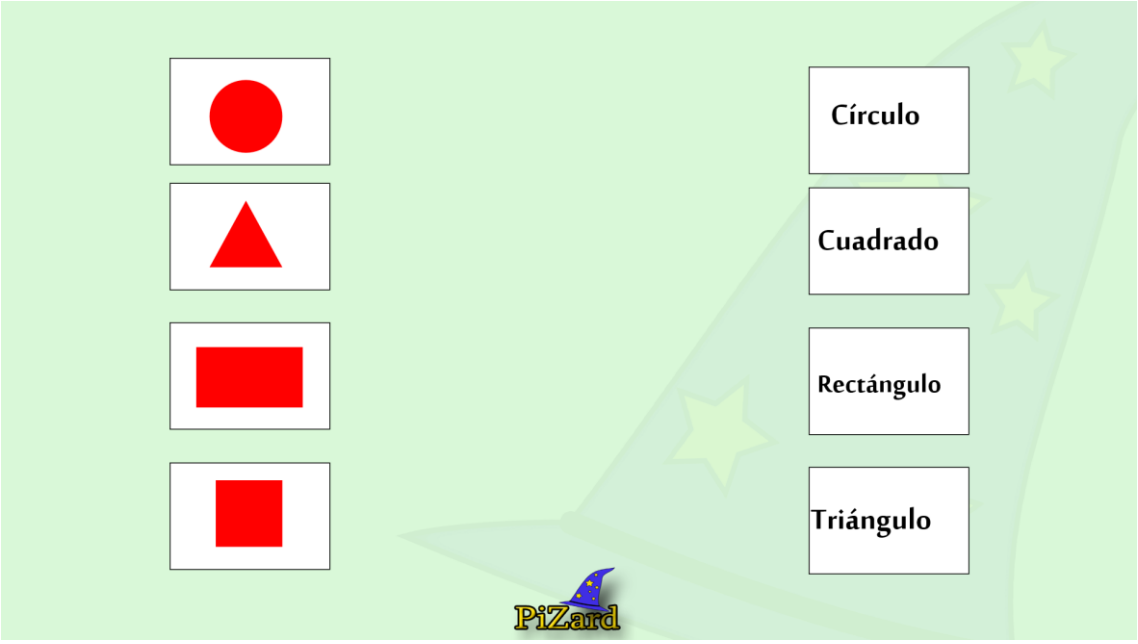


Figura 31: Diseño del template de relación o asociación

Este template está diseñado específicamente para una relación múltiple entre conceptos. Comparándolo con el anterior, este busca una relación en conjunto mientras que el template de pregunta busca la toma de una decisión en la elección de la respuesta correcta.

5. Conclusiones y trabajo futuro

5.1 Conclusiones

En este trabajo, se ha desarrollado y construido un prototipo de una plataforma de videojuegos adaptada a niños y jóvenes con necesidades de educación especial, que permita no solo jugar, sino también que sus padres y profesores creen juegos adaptados a sus necesidades, a los que se juega a través de una interfaz de botones adaptada.

Durante el desarrollo del proyecto, se ha investigado sobre los métodos educativos usados en la educación especial, especialmente el método TEACCH, y también se han adquirido conocimientos, tanto del lenguaje python, como de las tecnologías de desarrollo web utilizadas y sobre el desarrollo de hardware y diseño de PCBs.

Con la conclusión satisfactoria de un prototipo funcional, se ha establecido la prueba de concepto del proyecto. Con el prototipo, no pueden cubrirse todas las necesidades que se especificaban al comienzo de la fase de diseño, sin embargo, puede significar el inicio de una evolución hacia un sistema que si lo haga.

5.2 Trabajo futuro

Tras el desarrollo del trabajo, surgen una serie de mejoras que podrían efectuarse al sistema:

- Se podría implementar el template de relación, que ha sido diseñado pero no programado para este prototipo.

- Se podría haber aumentado y mejorado el banco de recursos por defecto con el que cuenta la aplicación.

- Algunos de los mecanismos del API de expansión del proyecto son complicados, simplificarlos puede ayudar a los programadores a implementar expansiones para el proyecto.

- El trabajo más importante que aún queda por hacer, es realizar pruebas en colegios e institutos con alumnos y profesores, tanto pruebas de integración y funcionamiento con alumnos, como de usabilidad con padres y profesores.

Referencias

[1] Gary B. Mesibov, Victoria Shea, Eric Shopler, Lynn Adams, Elif Merkler, Sloane Burgess, MAtt Mosconi, S. Michael Chapman, Christine Tanner, Mary E. Van Bourgondien, The TEACCH approach to autism spectrum disorders, Springer, 2004, pp. 1-12.

[2] Schopler, E. and Reichler, R., Parents as cotherapists in the treatment of psychotic children. Journal of Autism and Childhood Schizophrenia 1, Springer, 1971.

[3] Eric Shcopler, Gary B. Mesibov, The TEACCH Communication Curriculum, Springer, 1985.

[4] Patricia Howlin, Simon Baron-Cohen, Julie Hadwin, Enseñar a los niños autistas a comprender a los demás. Guía práctica para educadores, Ediciones Ceac, 2006.

[5] Sraff and Parents of division TEACCH, Teaching Activities for Autistic Children: Individualized and Treatment for Autistic and Developmentally Disabled Children, Proed, 1983.

[6] Susan Unok Marks, Jennifer Shaw-Hegwer, Carl Schrader, Tricia Longaker, Iris Peters, Fran Powers, Mark Levine, «Instructional Management Tips for Teachers of Students With Autism Spectrum Disorder (ASD),» *TEACHING Exceptional Children,* 2011.

[7] K. Jaegers, XNA 4.0 Game Development by Example: Beginner's Guide – Visual Basic Edition, Packt, 2011.

[8] L. Ullman, PHP Advanced and Object-oriented Programming, Pearson, 2012.

[9] Jonathan Chaffer, Karl Swedberg, Learning jQuery (Fourth Edition), Packt, 2013.

[10] Hasin Hayder, Joao Prado Maia, Lucian Gheorghe, Smarty PHP Template Programming and Applications, Packt, 2006.

[11] Magnus Lie Hetland, Beginning Python: From Novice to Professional, Apress, 2008.

[12] David Powers, PHP Solutions: Dynamic Web Design Made Easy, Apress, 2010.

[13] Al Sweigart, Making Games with Python & Pygame, http://inventwithpython.com (free book), 2012.

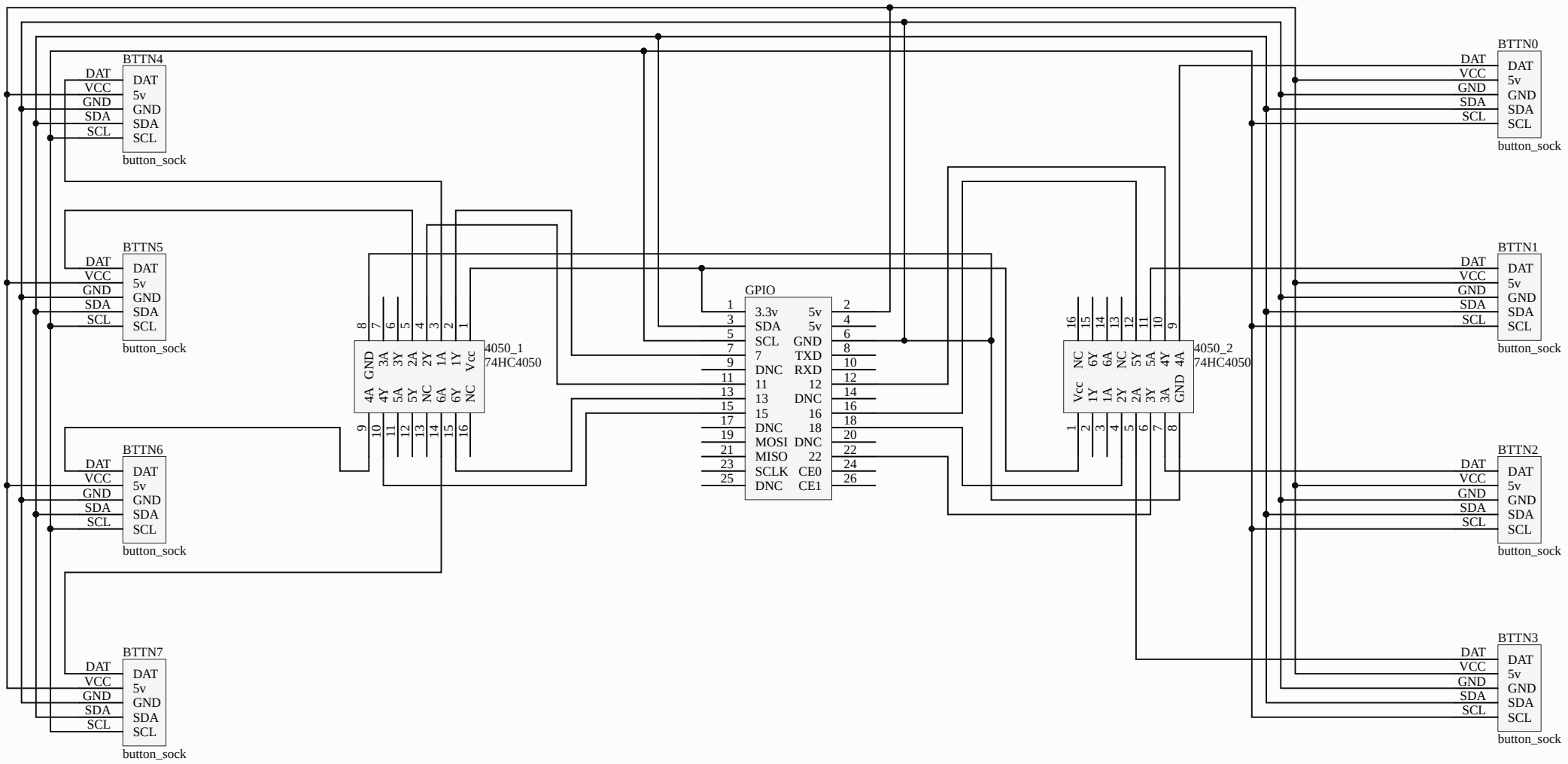
[14] Matt Richardson, Shawn Wallace, Getting Started with Raspberry Pi, O'Reilly Media, 2012.

[15] Simon Monk, Programming the Raspberry Pi: Getting Started with Python, McGraw-Hill/TAB Electronics, 2013.

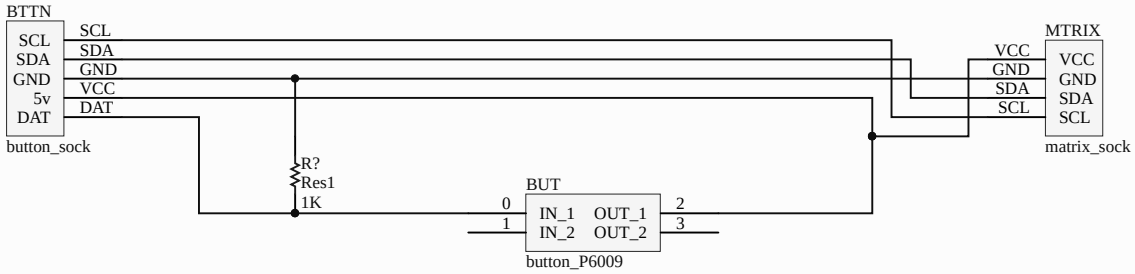
Glosario de acónimos

HDMI	High-Definition Multimedia Interface
LED	Light-Emitting Diode
API	Application Programming Interface
TEACCH	Treatment and Education of Autistic and Related Communication Handicapped Children
GPIO	General-Purpose Input/Output
SBC	Single Board Computer
RAM	Random Access Memory
I/O	Input/Output
SoC	System on a Chip
CSS	Cascading Style Sheets
AJAX	Asynchronous JavaScript and XML
PHP	PHP: Hipertext Protocol (recursivo)
USB	Universal Media Bus
GPU	Graphics Processing Unit
CPU	Central Processing Unit
I2C	Inter-Integrated Circuit
PCB	Printed Circuit Board
JSON	JavaScript Object Notation
HTML	HyperText Markup Language
LXDE	Lightweight X11 Desktop Environment
ESSID	Extended Service Set IDentifier
DHCP	Dynamic Host Configuration Protocol

Anexo: Esquemáticos PCB



Title	PiZard button expansion card for Raspberry Pi	
Size: A4	Number	Revision: 1.0
Date: 24/06/2014		Sheet of
File: D:\TFG\GPIO_adapter.SchDoc		Drawn By: Guillermo Climent



Title	PiZard button card for Raspberry Pi		
Size: A4	Number:		Revision: 1.0
Date: 24/06/2014		Sheet of	
File: D:\TFG\Button.SchDoc		Drawn By: Guillermo Climent	